

A Taxonomy of Parallel VHDL Simulation Techniques (Gee, Look at all the Dissertations Waiting Out There)

Gregory D. Peterson
Computer Aided Design Engineer
Wright Laboratories
United States Air Force
Dayton, Ohio
gdp@el.wpafb.af.mil

John C. Willis
Senior Engineer / Scientist
IBM Systems Technology &
Architecture Division
Rochester, MN
jwillis@acm.org

1 Abstract

Parallel VHDL simulation has emerged from solely being a subject of academic interest into an area of practical utility. This paper presents a taxonomy of parallel VHDL simulation techniques as they are actually manifest in a variety of parallel VHDL simulators. The IEEE DASC group on parallel VHDL simulation is using an evolution of this taxonomy as a key to simulator-specific recommended VHDL coding practices for efficient, high-performance parallel VHDL simulation.

2 Introduction

Over the last several years, VHDL has emerged as a powerful and important tool for developing and maintaining digital systems. Given its expressibility, VHDL is useful in describing the function of designs ranging from small logic circuits to large, abstract models of computer systems. One unfortunate fact of life in electronic design automation is the need for simulation speed and capacity almost always exceeds what is available.

In order to cost-effectively boost the performance of VHDL simulation, parallel simulation techniques can be adopted. Numerous groups have or are developing parallel VHDL simulators. Exploiting parallel processing techniques to improve performance is particularly promising because VHDL explicitly and consistently supports the notion of concurrent processes.

To help understand the challenges and opportunities in parallel VHDL simulation, the IEEE Design Automation Standards Committee (DASC) has established a parallel simulation study group chaired by John Willis (working group PAR applied for). The group operates a reflector, parallel@vhdl.org. For further information, mail parallel-request@vhdl.org or jwillis@acm.org.

The group is developing a bibliography detailing related research to help facilitate the development of recommended practices. This paper describes a taxonomy of parallel VHDL simulation techniques to help provide a structure for the bibliography and recommended practices. We can place research efforts and commercial products into this context to understand them, with appropriate expectations and recommended practices [WP95].

This paper includes issues related to the choice of parallel computer architecture for executing the VHDL simulator (Section 2), parallel simulation issues unique to VHDL (Section 3), conservative simulation protocols (Section 4), and optimistic simulation protocols (Section 5). We focus on parallel VHDL simulation in this paper rather than on the broader field of parallel discrete event simulation (PDES). For a general survey of PDES, Bailey, Briner, and Chamberlain [BBC94] and Fujimoto [RF90] give excellent overviews.

3 Architecture:

Parallel VHDL simulation has been implemented or can be envisioned on a number of different types of parallel architectures [MF66]. For purposes of this

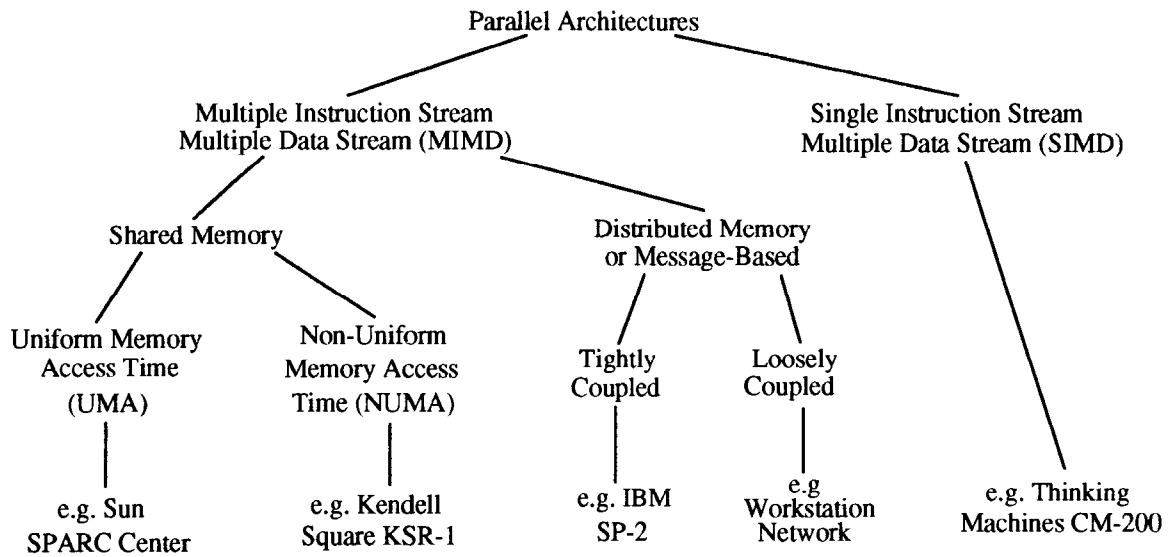


Figure 1 Parallel Architecture Taxonomy

discussion, we will generalize parallel architectures to include any machine with more than one data stream and the ability for communications between the data streams.

Figure 1 summarizes a popular taxonomy of parallel architectures along with illustrative examples of machines at each terminal branch of the taxonomy. It is illustrative that parallel VHDL simulators have *already* been run on each of these example architectures; parallel VHDL simulation is a reality.

Flynn [MF66] suggests the top-most branch in the taxonomy based on the number of independent instruction streams. Machines with multiple instruction streams are the more flexible and thus account for most of the interest in parallel VHDL simulation.

Within the MIMD branch, both shared memory and distributed (or message-based) machines are popular. Shared memory machines generally support 2 to about 20 processors whereas distributed memory systems can accommodate from several to hundreds of processors. The degree of coupling (expressed in terms of communication latency, bandwidth and affinity) ranges from uniform memory access time machines (small, tightly coupled) to large but loosely coupled distributed memory systems.

SIMD machines can be used to simulate VHDL models with a very small range of process data flows and component-specific behavior, as is often the case with gate-level models. Such machines often have high raw performance and hundreds to tens of thousands of processors. The processors in SIMD machines tend to be very simple and thus cost-effective.

4 VHDL-Specific Concerns for PDES:

VHDL is the most flexible and expressive language generally used to describe models for parallel, discrete event (PDES) simulation. As a result, VHDL introduces a set of challenges and opportunities seldom found in the more general body of PDES work. This section emphasizes these aspects of VHDL.

Signals are VHDL's primary mechanism for communicating between sequential processes. This correspondence between VHDL and the communicating sequential processes paradigm greatly facilitates compilation of VHDL for parallel machines by cleanly decoupling parallel and sequential semantics.

Signal drivers can project pending events (waveform elements) out into the temporal future. Mechanisms for merging pending events with new events

(representing inertial and transport delay) further confound interprocessor communication. In the most general case, messages cannot be sent when they are generated by a signal assignment; this is only possible under special case optimization.

VHDL signals have at least a delta (non-zero) propagation delay. Thus there are no zero-delay cycles which would serve to complicate conservative PDES. However in VHDL-93, shared variables were added, resulting in the potential for zero-delay communications involving two or more processes. This complication not only requires conservative PDES implementations to deal with deadlock induced by the simulation algorithm, it also leads to potentially non-deterministic results on almost any parallel VHDL implementation. Recognizing this as a problem, IEEE's RevCon (Review Committee) directed DASC to develop a solution. DASC's shared variable working group is refining a form of Ada's protected type as a partial solution.

VHDL's strong typing system is a very significant asset when simulating on parallel computers with heterogeneous instruction set architectures. Many hardware description languages do not specify aspects such as actual data type precision and correspondence of integers to bit vectors. When executed on a heterogeneous system, arbitrary mapping decisions between processes and processors have the potential for generating a wide variety of results.

Following the Ada model, VHDL's type system provides a very restricted form of pointer, known as an access type. Operators on access types are very limited, reducing the aliasing problems often encountered when parallelizing languages such as C (with more flexible pointers). VHDL-93's ability to create shared variables of access type complicate static aliasing analysis. Again, DASC's Shared Variable Working Group has already recognized the problem and is drafting a solution based on protected types.

By enabling design at a higher level of abstraction, VHDL has the ability to describe a system with fewer sequential processes than an equivalent gate-level design. Generally higher levels of abstraction simulate more efficiently (on a uniprocessor or parallel processor). However if too few processes are ready for evaluation within a given time window, a parallel simulator may need to idle processors. Research has addressed this problem by both

increasing the time window across which processes may be concurrently evaluated [CV91] and permitting concurrent evaluation of the same process [JW92].

5 Conservative algorithms:

In a conservative PDES, simulation activity always corresponds to activity which must occur in the actual system being simulated. In the simulation domain, time monotonically advances. Each process evaluation and signal propagation contributes to the simulation. Conceptually, conservative PDES is simple to implement and uses computations efficiently.

Exploitable parallelism is the greatest challenge to the use of conservative PDES for practical VHDL simulation. *Exploitable parallelism can be quantitatively expressed as the time-averaged duty cycle of all processors assigned to accomplish a parallel simulation.* For a conservative PDES implementation and a fixed number of processors, both the absolute performance and cost-efficiency are proportional to the exploitable parallelism.

Synchronous approaches to conservative PDES require that all processors complete execution of computations at a given simulation time before any may proceed on to executions at the next time step (in practice, pipelining across this time barrier is feasible). In effect, for synchronous, conservative PDES, all processors synchronize at a barrier before advancing to the next time step.

The exploitable parallelism of synchronous, conservative PDES implementations is critically dependent on both the average number of active processes at each instant in time and the uniformity with which these processes can be partitioned among processors. Several studies have used event logs and sampling of a central event queue to characterize the average number of processes active at any instant in time. For event driven languages (e.g. VHDL) and delay-based modeling styles (e.g. Vital), these studies tend to show a high activity variance across time (a few active processes at one instant, hundreds to millions at others).

Research at the University of Michigan showed that even relatively simple, but massive, SIMD machines can be effectively used for synchronous PDES if the domain of primitive process types and timing is suitably restricted [AD92].

Vantage's SpeedWave MT VHDL simulator [HD94] uses shared memory (UMA architecture) and a lockable, shared event queue so as to dynamically assign evaluations to processors. Since most VHDL models have at least several active processes at a time step (or none at all), we would expect SpeedWave MT to yield good speedups for the relatively small number of processors (2 to 8) typically configured around a shared memory. These are indeed the kind of encouraging results reported for SpeedWave MT.

Zycad's ViP (now withdrawn from Zycad's formal product line) [VV94] harnesses up to 64 tightly coupled MIPS processors and HDL-specific hardware assists to implement a synchronous, conservative PDES (tightly-coupled, message-based architecture). ViP uses dedicated hardware to pipeline the VHDL simulation cycle. Static partitioning of VHDL processes to processors eliminates contention for central event queues and improves cache performance. However, we might expect ViP to be more sensitive to how evenly process activity is distributed across processors at each simulation cycle (due to static partitioning). Zycad users indeed report encouraging speedups using several tens of processors, however substantial effort needs to be invested in suitable coding styles and partitioning.

Since the total number of processes active at any instant in time is inherently bounded for a given model, stimulus and time instant, further increases in simulation performance may be gained by allowing each processor participating in the simulation to execute evaluations at one or more instants in time which may be distinct for each processor. Provided these evaluations all correspond to the actual system under simulation, we will refer to this approach as *asynchronous, conservative PDES*.

Several projects have studied the natural parallelism inherent in VHDL models as a result of the target machine, simulation algorithm and natural parallelism of the model [TC92, YH92, YH93, JS93, PA94]. These studies are essential to improve our understanding of the constraints governing the absolute performance of a simulation

The US Air Force Institute of Technology (AFIT) [KK95] has developed an asynchronous, conservative PDS simulator following the Bryant/Chandy/Misra algorithm [RB77]. AFIT's research

focuses on optimization of null-messages and partitioning algorithms to effectively use Intel iPSC/2 hypercubes and Paragons.

The Auriga compiler, involving Carnegie Mellon University [JW91], The University of Minnesota [JW95], IBM [JW93] and FTL Systems, uses an asynchronous, conservative approach to PDES. During compilation, Auriga clusters elaborated VHDL processes (or process fragments) which may execute at the same instant in time (reduces to cycle-driven approach in special cases) [JW91]. These clusters retain their own local clock, which advances monotonically forward as simulation progresses. In order to increase the acceptable variance between local clocks, each event message not only implies the time at which an event occurs, but also the minimum time until the next event on that signal or sub-signal. The minimum time until the next event is largely based on a compile-time dataflow analysis. Using this approach, substantial speedups are feasible using systems with tens to hundreds of message-linked processors (e.g. Thinking Machines CM-5, Intel Paragon and IBM SP2), min-cut like static partitioning and low communication latency [JW95].

Using trace-driven simulation, researchers at the University of Genoa also concluded that asynchrony, correct static partitioning, and tight coupling were critical for effective parallel VHDL simulation [CG94]. Static partitioning needs to take into account both the computational complexity of each process and the predicted event traffic. Tight coupling seems to operationally mean a communications latency no more than about an order of magnitude larger than the average time to execute a process from one wait statement to the next.

Several parallel VHDL simulator projects, which expect to use conservative PDES, are in the design stages including a product development at TGI [SO95] and a research project at the University of Arizona [SG95]. Hopefully their developers will share experiences at future VIUF conferences.

6 Optimistic Algorithms:

In contrast to conservative simulation, optimistic simulation allows simulation time on each processor to advance non-monotonically. Optimistic parallel simulation protocols include some possibility of a non-causal message arriving resulting in a "rollback" to a previous state [DJ85]. A simulator can either

immediately rollback and proceed with the simulation, or it can check to see if the non-causal message resulted in any incorrect computations. If so, the simulator rolls back; otherwise it proceeds with the simulation. When a rollback does occur, the state is returned to the last saved state immediately prior to the non-causal message received. The local clock is rolled back to the time of the prior state and work previously performed beyond this point in simulated time is discarded as incorrect. Incorrect messages sent out must also be discarded, which can result in cascading rollbacks and performance degradation.

Optimistic simulators can use either aggressive cancellation, in which all incorrect messages are immediately tracked down and discarded via "antimessages", or by lazy cancellation, in which messages are only discarded after a rollback when the messages are known to be incorrect. Lazy cancellation can improve performance by easing the number of required rollbacks (since some messages may be correct and not require cancellation). If erroneous messages are not cancelled immediately; however, significant work may need to be discarded once the messages are eventually determined to be incorrect. Rajan and Wilsey attempt to exploit the benefits and ameliorate the penalties of each of these techniques by dynamically switching between aggressive and lazy cancellation over the course of a simulation as appropriate [RW95].

Applying throttling mechanisms to restrict the optimism is another technique to prevent excessive rollbacks and recomputation, or risk, allowed. Potential parallelism is sacrificed to reduce the penalties associated with rolling back state. A good example of this is Sokol's moving time window (MTW) protocol, in which local clocks are only allowed to advance beyond the global simulation time by a fixed amount of time. Steinman developed his breathing time buckets protocol to adaptively change the window size (to "breathe") based on the characteristics of the simulation. Research at the University of Cincinnati includes application of control theory to adapt to the dynamic conditions of VHDL simulation [PW93c, PW94].

In order to enable rollbacks, some state saving is required. An optimistic simulator can either save state incrementally with each change, or take periodic checkpoints of its state [FW95]. One can use analytic techniques to determine which approach is best [PW93a], or control theory can be employed

to develop adaptive algorithms for the best state saving frequency [PW93b]. The simulator can also either save state to support rollback on an element by element basis or it can save state so that all elements must rollback together. The cost of saving state information and rollbacks for individual elements is higher, but rollbacks can be restricted to only the elements affected by a non-causal event. In essence, the cost of saving state must be balanced against the penalty of rolling back in time. Costa and De Gloria find there is negligible performance degradation when the cost of rollbacks is less than 100 VHDL statements [CG94]. As the rollback cost increases, simulation performance degradation becomes significant.

While saving state, the memory requirements of optimistic algorithms can become excessive. Hence, "fossil collection" algorithms are used to determine when it is safe to reclaim memory. As the parallel simulation progresses, a lower bound on the value of the local clocks, the "global virtual time" (GVT), is periodically computed. No rollback can reset a local clock to a value below the GVT, so the saved state from before the GVT can be discarded and simulation results up to the GVT can be committed. Several algorithms exist for computing GVT and trade off the computation cost of time spent computing GVT with the benefit of reducing memory demands [DFW94].

Thus far, few results from optimistic parallel VHDL simulation exist. The largest commercial optimistic VHDL simulation effort to date was developed by Jade Simulations and Vantage, but no product was ever sold and no results are known to have been published.

In recent years, researchers at the University of Cincinnati and MTL Systems, Inc. developed the QUEST simulator [CV91] followed by the VAST simulator [MC94]. These simulators facilitated research into many of the issues discussed above, including lazy or aggressive cancellation, throttling mechanisms, and state saving [DFW94, FW95, PW93a, PW93b, PW93c, PW94, RW95]. The QUEST simulator was developed for distributed memory architectures and a version of Quest also exists for execution using MPI (thus supporting either a shared memory or distributed memory model, but not a mix). VAST evolved from QUEST to support threaded execution on shared memory, symmetric multiprocessors. Quest and VAST can both execute on a network of workstations connected

via Ethernet or SCRAMnet [MC92, MC94]. The QUEST simulator is freely available for noncommercial use through MTL Systems. IntellX, a subsidiary of MTL, is preparing to commercialize the VAST simulator.

Although parallel simulation protocols typically are either conservative or optimistic, Hamnes and Tripathi propose using adaptive protocols to create simulators that switch between conservative and optimistic at runtime [HT94]. This may prove to be the best approach for achieving high performance in parallel VHDL simulation, although, once again, research remains to determine which adaptive techniques are most appropriate for particular types of models and parallel architecture.

7 Conclusions:

Parallel VHDL simulation has been demonstrated on almost every widely available parallel computers and using numerous parallel simulation algorithms. By way of definition, this paper presented a taxonomy of parallel architectures, explained some of the implications VHDL has for PDES, then sketched a taxonomy of existing parallel VHDL simulators. The top level branch of this taxonomy divides conservative and optimistic algorithms. Within the conservative branch there are both synchronous and asynchronous subsets. An even wider range of taxonomy options are available among optimistic algorithms.

Using this taxonomy, simulation practitioners can delineate the differences between various approaches to parallelizing VHDL simulation. This allows developers to better understand their options and make more informed choices, helps researchers understand the current state of the art and the relevance of certain research efforts, provides a structure that is useful pedagogically, and gives an overview of the current state of the art in parallel VHDL simulation.

The DASC working group's parallel VHDL simulation bibliography will evolve over time within this framework, but can be inspected at any given time to give a snapshot of the current state of the art for parallel VHDL simulation. The bibliography developed by the DASC parallel simulation study group can use this taxonomy as a framework to make it easier to find papers relevant to a given parallel VHDL simulation approach. Parallel VHDL products available on the market can be classified to

enable customers to determine which best suit their needs. Finally, researchers can better understand the similarities and differences with published results to help determine trends and techniques to be incorporated into any recommended practices document.

8 Acknowledgments

The authors wish to acknowledge the IEEE DASC Parallel VHDL Group (over 100 people) for their contributions to this manuscript. Specific contributions describing simulators in this paper or other assistance came from Philip Wilsey, Praveen Chawla, John Hines, Sumit Ghosh, Paul Menchini, Thomas Hartrum, Hal Carter, Bill Paulsen, Zhiyuan Li, Serafin Olcoz, Bob Parker, Moon Jung Chung, Dave Ackley, Paul Satre, John Reed, Pam Fossey, Jeff Snyder, Vijay Vaidyanathan and others.

9 References

- [WP95] John Willis and William Paulsen, "Style Guidelines for Effective Use of Parallel and Multithreaded VHDL Simulators", Proceedings of the Fall 1995 VHDL International User's Forum, Conference Management Services, October 1995.
- [BBC94] Mary Bailey, Jack Briner and Roger D. Chamberlain, "Parallel Logic Simulation of VLSI Systems", Computing Surveys, Volume 26, Number 3, September, 1994, pages 255-294.
- [RF90] Richard M. Fujimoto, "Parallel Discrete Event Simulation", Communications of the ACM, Volume 33, Number 10, October, 1980, pages 30 - 53.
- [MF66] Michael J. Flynn, "Very High-Speed Computing Systems", Proceedings of the IEEE, Volume 54, Number 12, December, 1966, pages 1901-1909.
- [JW92] J. Willis and D. Siewiorek, "Optimizing VHDL Compilation for Parallel Simulation", IEEE Design and Test Magazine, Volume 9, Number 3, September, 1992, pages 42-53.
- [AD92] Alan D. Cabrera, Moon Jung Chung

- and Yunmo Chung, "A Parallel VHDL Simulator on the Connection Machine", VHDL International User's Forum, May, 1992, pages 83 - 93.
- [HD94] H. Dai and Bill Paulsen, "Multithreading VHDL Simulation", VHDL International User's Forum, Conference Management Services, November, 1994, pp. 4.33 - 4.38.
- [VV94] V. Vaidyanathan, Presentation on Zycad ViP to IEEE Design Automation Standards Study Group on Parallel Simulation, (copies available from SG chair), June, 1994.
- [TC92] T. Collette, "Architecture et validation comportementale en VHDL d'un calculateur parallèle dédié à la vision par ordinateur", INP Grenoble Thesis, Microelectronique, Sept. 1992.
- [YH92] Y. Herve, "Accélération maximum théorique d'une simulation répartie à partir de la trace d'une simulation séquentielle", Technique et Science Informatique (TSI), November, 1992.
- [YH93] Y. Herve, "Parallel Simulation: Self-Extraction of the Natural Parallelism of VHDL Models", VHDL International User's Forum, Conference Management Services, April, 1993, pages 287-296.
- [JS93] J. Sissler, "Assessing the Potential of Multi-Threaded VHDL Simulation", VHDL International User's Forum, Conference Management Services, October, 1993, pp 131-136.
- [PA94] Peter J. Ashenden, Henry Detmold, Wayne S. McKeen, "A Centralized Queue Parallel Simulator for VHDL", International Conference on Simulation and Hardware Description Languages (ICSHDL), Editors Philip A. Wilsey and David A. Rhodes, Society for Computer Simulation, January 1994, pages 161-166.
- [KK95] Kevin L. Kapp, Thomas C. Hartrum and Tom S. Wailes, "An Improved Cost Function for Static Partitioning of Parallel Circuit Simulations Using a Conservative Synchronization Protocol", Proceedings of the 9th Workshop on Parallel and Distributed Simulation (PADS '95), IEEE Computer Society Press, June, 1995.
- [RB77] R.E. Bryant, "Simulation of Packet Communication Architecture Computer Systems," Tech Report MIT/LCS/TR-188, Laboratory for Computer Science, MIT, Cambridge, Mass., 1977.
- [JW91] J. Willis, "Optimizing VHDL Compilation for Parallel Simulation", Ph.D. Thesis, Department of Electrical and Computer Engineering, Carnegie-Mellon University, Pittsburgh, 1991.
- [JW91a] John Willis and Daniel Siewiorek, "Discrete Interval Simulation of Hardware Description Languages using an Optimizing Compiler", 22nd Annual Pittsburgh Conference on Modeling and Simulation, The Society for Computer Simulation, May, 1991.
- [JW95] J. Willis, Z. Li and Tsang-Pu Lin, "Use of Embedded Scheduling to Compile VHDL for Effective Parallel Simulation", Proceedings of EuroDAC/EuroVHDL 1995.
- [JW93] John Willis, Rob Newshutz, Lance Thompson, Jeff Graves, Tom Dillinger, Jeff Snyder, Nimish Radia, Joe Skovira, David Blaauw, Sidhartha Mohanty, Zhiyuan Li, Sandra Samelson, and Matt Lin, "MinSim: Optimized, Compiled VHDL Simulation Using Networked and Parallel Computers", Proceedings of the VHDL International User Forum, Conference Management Services, October, 1993.
- [CG94] Alessandra Costa and Alessandro De Gloria, "An Evaluation System for Distributed-Time VHDL Simulation", Proceedings of the 8th Workshop on Parallel and Distributed Simulation (PADS), July, 1994, pages 147-150.
- [SO95] Serafin Olcoz, Communication to DASC Parallel VHDL SG, 1995.
- [SG95] Sumit Ghosh, Communication to DASC Parallel VHDL SG, 1995.

- [DJ85] David Jefferson, "Virtual Time", ACM Transactions on Programming Languages and Systems, Volume 7, Number 3, July 1985 pages 405-425.
- [RW95] R. Rajan and P.A. Wilsey, "Dynamically Switching Between Lazy and Aggressive Cancelation in a Time Warp Simulator", Proceedings of the 28th Annual Simulation Symposium, IEEE Computer Society Press, April, 1995.
- [FW95] J. Fleischman and P. A. Wilsey, "Comparative Analysis of Periodic State Saving Techniques in Time Warp Simulators", Proceedings of the 9th Workshop on Parallel and Distributed Simulation (PADS 95), June, 1995.
- [PW94] A.C. Palaniswamy and P.A. Wilsey, "Scheduling Time Warp Processes using Adaptive Control Techniques," 1994 Winter Simulation Conference, Edited by J.D. Tew, S. Manivannan, D.A. Sadowski and A.F. Seila, December, 1994, pages 731-738.
- [PW93a] A.C. Palaniswamy and P.A. Wilsey, "An Analytic Comparison of Periodic Checkpointing and Incremental State Saving", Proceedings of the 7th Workshop on Parallel and Distributed Simulation (PADS), Society for Computer Simulation, July, 1993, pages 127 - 134.
- [PW93b] A. Palaniswamy and P.A. Wilsey, "Adaptive Checkpoint Intervals in an Optimistically Synchronized Parallel Digital System Simulator", VLSI 93, September, 1993, pages 353-362.
- [PW93c] A.C. Palaniswamy and P.A. Wilsey, "Adaptive Bounded Time Windows in an Optimistically Synchronized Simulator", Third Great Lakes Symposium on VLSI, March, 1993, pages 114-118.
- [DFW94] L. M. D'Souza, X. Fan and P.A. Wilsey, "pGVT: An Algorithm for Accurate GVT Estimation", Proceedings of the 8th Workshop on Parallel and Distributed Simulation (PADS 94), Society for Computer Simulation, July, 1994, pages 102-109.
- [CV91] H. Carter, R. Vemuri, P.A. Wilsey, J. Aylor, R Waxman and T. Hartrum, "The Quest Project: High Speed Acceleration of VHDL Simulation, Synthesis and ATPG Using Distributed Processing", VHDL International User's Forum, Conference Management Services, April, 1991, pp 85-90.
- [MC94] T. McBrayer, V. Krishnaswamy, S. Mohanty, L. Moore, X. Liu, J. Carter, D. Charley, P.A. Wilsey, D.A. Hensgen, H.W. Carter, P. Chawla, J. Colier, S. Bilik, "VAST: TimeWarp Simulation of VHDL on SMP Workstations", VHDL International User's Forum, Conference Management Services, November, 1994, pp 4.17-4.32.
- [MC92] T. McBrayer, D. Charley, P.A. Wilsey, D.A. Hensgen, "A Parallel, Optimistically Synchronized VHDL Simulator Executing on a Network of Workstations", Proceedings of the Fall 1992 VHDL International User's Forum, October, 1992, pages 218-222.
- [HT94] Donald O. Hammes and Anand Tripathi, "Investigations in Adaptive Distributed Simulation", Society for Computer Simulation, July, 1994, pages 20-23.